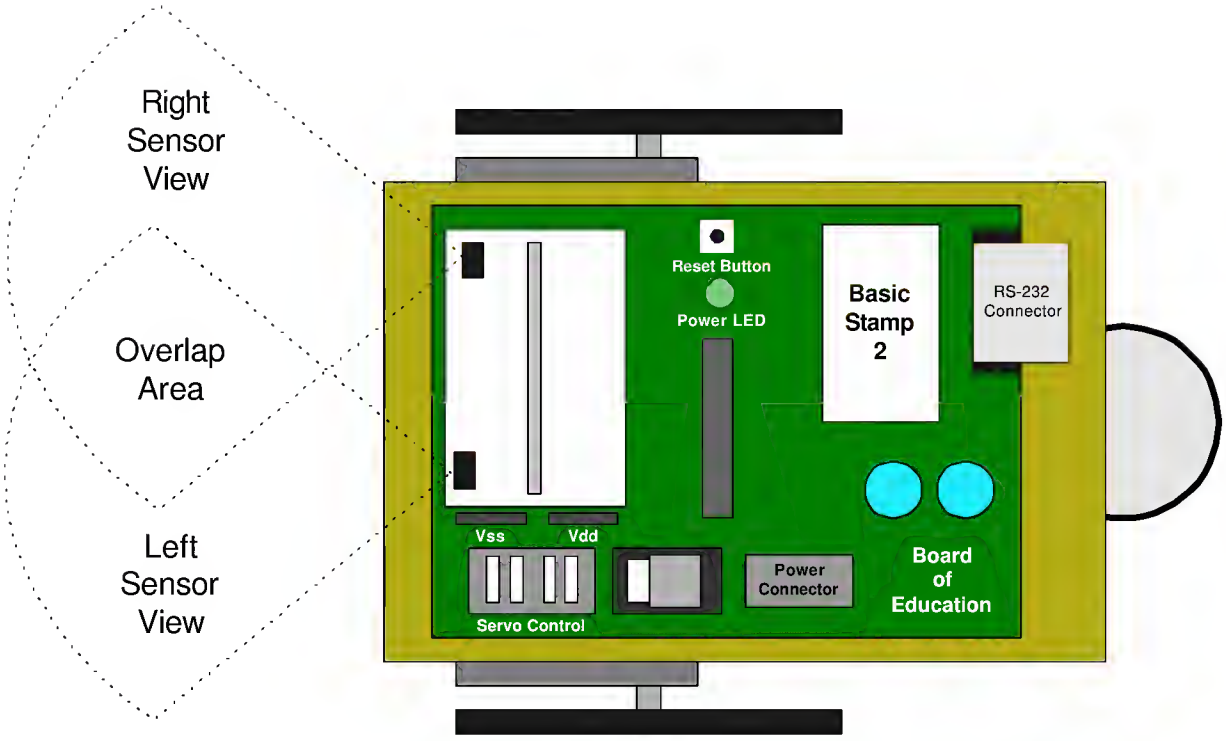
Maze Runners



Laura Wong

Table of Contents

List of Figures

List of Photographs

List of Tables

Code Listings

1. Problem

For a robot to self navigate from a starting point to an ending point in a maze

2. Hypothesis

The robot will successfully travel through the maze using sensors and at different speeds.

3. Abstract

In the experiment, two different robots are programmed to navigate through simple mazes using sensors to see the wall. Two algorithms are used: the Random Walk and the Right Hand Rule. The Random Walk allows the robot to walk around freely in the maze. The Right Hand Rule has the robot follow the right wall until it gets out of the maze. A simple maze has a wall made up of only one surface. A complex maze is a maze that is made up of two or more surfaces.

The two robots used in experiment are the WOA II and the Boe-Bot. The WAO II uses force sensors that must contact the wall. The WAO II contains a 24-step programming memory, limiting the amount of steps in a program. The Boe-Bot uses infrared sensors and has a 2048 bytes memory. It also executes more complex programs.

The Random Walk algorithm is expected to get the robot out of the maze only sometimes. The Right Hand Rule is expected to get the robot out of any simple maze. It may not get be sufficient to get the robot out of a complex maze. The robots will be tested for speed, accuracy, and repeatability.

4. Research

The idea of a robot is not new. For thousands of years man has been imagining intelligent mechanized devices that perform human-like tasks. He has built automatic toys and mechanisms and imagined robots in drawings, books, plays and science fiction movies.

In fact, the term "robot" was first used in 1920 in a play called "R.U.R." or "Rossum's Universal Robots" by the Czech writer Karel Capek. The plot was simple: man makes robot then robot kills man! Many movies that followed continued to show robots as harmful, menacing machines.

More recent movies, however, like the 1977 "Star Wars", portray robots such as "C3PO" and "R2D2" as man's helpers. "Number Five" in the movie "Short Circuit" and C3PO actually take on a human appearance. Robots which are made to look and act like human are called "androids".

In 1941, science fiction writer Isaac Asimov first used the word "robotics" to describe the technology of robots and predicted the rise of a powerful robot industry. His prediction has come true. Recently there has been explosive growth in the development and use of industrial robots to the extent that terms like "robot revolution", "robot age", and "robot era" are used. "Robotics" is now an accepted word which describes all technologies associated with robots.

In 1956, George Devil and Joseph Engelberger formed the world's first robot company. Devol predicted that the industrial robot would "help the factory operator in a way that can be compared to business machines as an aid to the office worker". A few years later, in 1961, the very first industrial robot was "employed" in a General Motors automobile factory in New Jersey. Since 1980, there has been an expansion of industrial robots into non-automotive industries.

Fully functioning androids are many years away due to the many problems that must be solved. However, real, working, sophisticated robots are in use today and they are revolutionizing the workplace. These robots do not resemble the romantic android concept of robots. They are industrial manipulators and are really computer controlled

"arms and hands". Industrial robots are so different to the popular image that it would be easy for the average person not to recognize one.

Robots offer specific benefits to workers, industries and countries. If introduced correctly, industrial robots can improve the quality of life by freeing workers from dirty, boring, dangerous and heavy labor. It is true that robots can cause unemployment by replacing human workers, but robots also create jobs: robot technicians, salesmen, engineers, programmers and supervisors. The benefits of robots to industry include improved management control and productivity and consistently high quality products. Industrial robots can work tirelessly night and day on an assembly line without an loss in performance. Consequently, they can greatly reduce the costs of manufactured goods. As a result of these industrial benefits, countries that effectively use robots in their industries will have an economic advantage on the world market.

4.1 History of Computers and Robots

A brief review of robot development is important because it puts the current interest in them into a historical perspective. The following theme is the list of dates highlight the growth of automated machines, which led to the development of the industrial robots currently available.

In 1801 Joseph Jacquard invents a textile machine which is operated by punch cards. The machine is called a “programmable loom” and goes into mass production. Twenty-nine years later,1830, American Christopher Spencer designs a cam-operated lathe. In 1892 the United States, Seward Babbitt designs a motorized crane with gripper to remove ingots from a furnace. 1921 was the year the first reference to the word robot appears in a play opening in London. The play, written by Czechoslovakian Karel Capek, introduces the word robot from the Czech robota, which means a serf or one in subservient labor.

In 1938 Americans Willard Pollard and Harold Roselund design a programmable paint-spraying mechanism for the DeVilbiss Company. Eight years later George Devol patents a general-purpose playback device for controlling machines. The device uses a magnetic process recorder. In the same year the computer emerges for the first time.

American scientists J. Presper Eckert and John Mauchly build the first large electronic computer called the Eniac at the University of Pennsylvania.

A second computer, the first general-purpose digital computer, dubbed Whirlwind, solves its first problem at M.I.T. In 1948, Norbert Wiener, a professor at M.I.T., publishes Cybernetics, a book which describes the concept of communications and control in electronic, mechanical, and biological systems. 1951 was the year a teleoperator-equipped articulated arm is designed by Raymond Goertz for the Atomic Energy Commission. Three years following this design the first programmable robot is designed by George Devol, who coins the term Universal Automation. He later shortens this to Unimation, which becomes the name of the first robot company. 1959 Planet Corporation markets the first commercially available robot.

A year later, the Planet Corporation Unimation is purchased by Condec Corporation and development of Unimate Robot Systems begins. American Machine and Foundry, later known as AMF Corporation, markets a robot, called the Versatran. In 1962, General Motors installs the first industrial robot on a production line. The robot selected is a Unimate. In the year 1964, artificial intelligence research laboratories are opened at M.I.T., Stanford Research Institute (SRI), Stanford University, and the University of Edinburgh. 1968 SRI builds and tests a mobile robot with vision capability, called Shakey. In 1970 At Stanford University a robot arm is developed which becomes a standard for research projects. The arm is electrically powered and becomes known as the Stanford Arm. 1973 was the year the first commercially available minicomputer-controlled industrial robot is developed by Richard Hohn for Cincinnati Milacron Corporation. The robot is called the T3, The Tomorrow Tool.

In 1974, Professor Scheinman, the developer of the Stanford Arm, forms Vicarm Inc. to market a version of the arm for industrial applications. The new arm is controlled by a minicomputer. In 1976 Robot arms are used on Viking 1 and 2 space probes. Vicarm Inc. incorporates a microcomputer into the Vicarm design. The year 1977 ASEA, a European robot company, offers two sizes of electric powered industrial robots. Both robots use a microcomputer controller for programming and operation. In the same year Unimation purchases Vicarm Inc. In 1978 the Puma (Programmable Universal Machine for Assembly) robot is developed by Unimation from Vicarm techniques and with

support from General Motors. 1980 was the year the robot industry starts its rapid growth, with a new robot or company entering the market every month.

In 1954, George C. Devol filed a U.S. patent for a programmable method for transferring articles between different parts of a factory, he wrote :
"The present invention makes available for the first time a more or less general purpose machine that has universal application to a vast diversity of applications where cyclic control is desired." This quote is referring to robots that can transfer objects from one end to a factory to another without a remote control.

In 1956, Devol met Joseph F. Engelberger, a young engineer in the aerospace industry. With others, they set up the world's first robot company, Unimation, Inc., and built their first machine in 1958. Their initiative was a great deal ahead of its time; according to Engelberger, Unimation did not show a profit until 1975.

The first industrial robot saw service in 1962 in a car factory run by General Motors in Trenton, New Jersey. The robot lifted hot pieces of metal from a die-casting machine and stacked them.

Japan, by comparison, imported its first industrial robot from AMF in 1967, at which time the United States was a good 10 years ahead of Japan in robotics technology. The enormous effort put forth by Japanese industry is best evidenced by the fact that Unimation was eventually reduced to handing over its pioneering robot technology to Kawasaki Heavy Industries in a licensing deal in 1968.

The production of robots in comparison between the United States and Japan differed from each other through the amount of companies within its boundaries.

By 1990, there were more than 40 Japanese companies, including giants like Hitachi and Mitsubishi, which were producing commercial robots. By comparison, there were approximately one dozen U.S. firms, led by Cincinnati Milacron and Westinghouse's Unimation. In 1979, the U.S. leader, Unimation, was the only company in the world actively marketing an advanced assembly robot.

In 1982, GM, the largest single user of robots in the world, signed a pact with Fanuc Ltd. for a joint robotics venture to make and market robots in the United States. In the first six months of operation, more than half of the 100 robots sold by the joint

venture went to GM, locking out other U.S. companies from the largest single buyer in the market. As of 1987 the U.S. market was valued at more than $170 billion.

Robotics is the field of computer science and engineering concerned with creating robots, devices that can move and react to sensory input. Robotics can also deal with a branch of computer science called Artificial Intelligence. Robots are now widely used in factories to perform high-precision jobs such as welding and riveting. They are also used in special situations that would be dangerous for humans. In cleaning toxic wastes or defusing bombs. Although great advances have been made in the field of robotics during the last decade, robots are still not very useful in everyday life consumer products, as they are too clumsy to perform ordinary household chores.

Law	Description
1	A robot may not injure a human being, or, through inaction, allow a human being to come to harm.
2	A robot must obey the orders given it by human beings except where such orders would conflict with the First Law.
3	A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

Table 1. Asimov's Three Laws of Robotics

The term *robotics* was introduced by writer Isaac Asimov. In his science fiction book, *I, Robot,* published in 1950, he presented Three Laws of Robotics.

4.2 Robot Technology

Algorithms are the systematic procedure that produces--in a finite number of steps--the answer to a question or the solution of a problem. The name derives from the Latin translation, *Algoritmi de numero Indorum,* of the 9th-century Muslim mathematician al-Khwarizmi's arithmetic treatise "Al-Khwarizmi Concerning the Hindu Art of Reckoning."

For questions or problems with only a finite set of cases or values an algorithm always exists (at least in principle); it consists of a table of values of the answers. In general, it is not such a trivial procedure to answer questions or problems that have an infinite number of cases or values to consider, such as "Is the natural number (1, 2, 3, . . .) *a* prime?" or "What is the greatest common divisor of the natural numbers *a* and *b*?" The first of these questions belongs to a class called decidable; an algorithm that produces a yes or no answer is called a decision procedure. The second question belongs to a class called computable; an algorithm that leads to a specific number answer is called a computation procedure.

Artificial Intelligence is the capacity of a digital computer or computer-controlled robot device to perform tasks commonly associated with the higher intellectual processes characteristic of humans, such as the ability to reason, discover meaning, generalize, or learn from past experience. The term is also frequently applied to that branch of computer science concerned with the development of systems endowed with such capabilities. Research on artificial intelligence began soon after the development of the modern digital computer in the 1940s. Early investigators quickly recognized the potential of computing devices as a means of automating thought processes. Over the years, it has been demonstrated that computers can be programmed to carry out very complex tasks--as, for example, discovering proofs for theorems or playing chess--with great proficiency.

High-level programming languages, while simple compared to human languages, are more complex than the languages the computer actually understands, called machine languages. Each different type of CPU has its own unique machine language.

A vocabulary and set of grammatical rules for instructing a computer to perform specific tasks. The term programming language usually refers to high-level languages, such as BASIC, C, C++, COBOL, FORTRAN, Ada, and Pascal. Each language has a unique set of keywords (words that it understands) and a special syntax for organizing program instructions.

4.2.1 Computer Algorithms

Algorithms are also known as a branch of computer science in which machines are designed to carry out tasks that require step-by-step reasoning processes. Examples are computers that can understand speech, recognizing shapes, solve problems, and learn from previous work. It was first studied in the 1950's and then developed in the 1970's and 1980's, artificial intelligence has been used to perform medical diagnoses, locate new mineral deposits in the earth, reason legal cases and control robots in manufacturing plants.

A formula or set of steps for solving a particular problem. To be an algorithm, a set of rules must be unambiguous and have a clear stopping point. Algorithms can be expressed in any language, from natural languages like English or French to programming languages like FORTRAN. We use algorithms every day. For example, a recipe for baking a cake is an algorithm. Most programs, with the exception of some artificial intelligence applications, consist of algorithms. Inventing elegant algorithms -- algorithms that are simple and require the fewest steps possible -- is one of the principal challenges in programming.

The definition of algorithms is generally left up to an implementation. However, this architecture defines two critically important algorithms. One is related to pad mobility, and the other is concerned with maintaining resemblance in data streams when servicing a control request and ensuring reliability of control messages regardless of the low-level communication protocol in use.

4.2.1.1 Declarative Style

4-oz. chocolate	3 eggs
1 cup butter	1 tsp. vanilla
2 cups sugar	1 cup flour

Melt chocolate and butter. Stir sugar into melted chocolate. Stir in eggs and vanilla. Mix in flour. Spread mix in greased pan. Bake at 350 degrees for 40 minutes or until inserted fork comes out almost clean. Cool in pan before eating.

Figure 1. Chocolate Cake Recipe

4.2.1.2 Procedural Style

```
Declare variables
        chocolate       eggs        mix
        butter          vanilla     fork
        sugar           flour

mix = melted ((4 * chocolate) + butter)
mix = stir ( mix + (2 * sugar))
mix = stir ( mix + (3 * eggs) + vanilla)
mix = stir ( mix + flour)
spread ( mix )
While not clean ( fork )
        bake ( mix, 350 )
```

Figure 2. Chocolate Cake Psuedo Program Code

4.2.1.3 Flow Charts

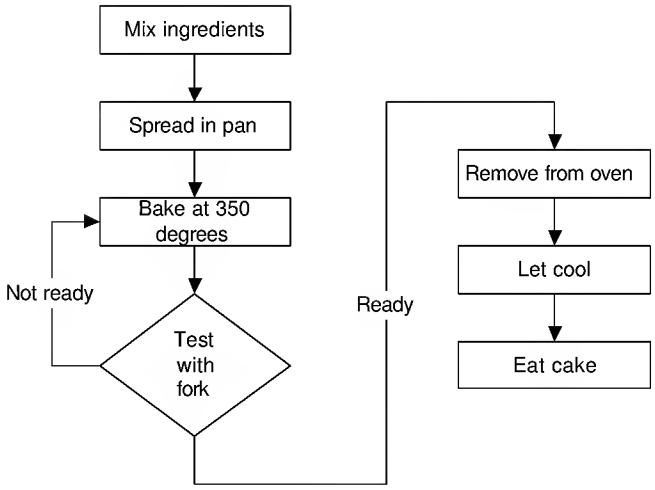


Figure 3. Flow chart to bake a chocolate cake

4.2.2 Artificial intelligence

Artificial Intelligence (AI) is the area of computer science focusing on creating machines that can engage on behaviors that humans consider intelligent. The ability to create intelligent machines has intrigued humans since ancient times, and today with the advent of the computer and 50 years of research into AI programming techniques, the dream of smart machines is becoming a reality. Researchers are creating systems which can mimic human thought, understand speech, beat the best human chess player, and countless other feats never before possible.

Artificial Intelligence, or AI for short, is a combination of computer science, physiology, and philosophy. AI is a broad topic, consisting of different fields, from

machine vision to expert systems. The element that the fields of AI have in common is the creation of machines that can "think".

In order to classify machines as "thinking", it is necessary to define intelligence. To what degree does intelligence consist of for example, solving complex problems, or making generalizations and relationships? And what about perception and comprehension? Research into the areas of learning, of language, and of sensory perception have aided scientists in building intelligent machines. One of the most challenging approaches facing experts is building systems that mimic the behavior of the human brain, made up of billions of neurons, and arguably the most complex matter in the universe. Perhaps the best way to gauge the intelligence of a machine is British computer scientist Alan Turing's test. He stated that a computer would deserves to be called intelligent if it could deceive a human into believing that it was human.

Artificial Intelligence has come a long way from its early roots, driven by dedicated researchers. The beginnings of AI reach back before electronics, to philosophers and mathematicians such as Boole and others theorizing on principles that were used as the foundation of AI Logic. AI really began to intrigue researchers with the invention of the computer in 1943. The technology was finally available, or so it seemed, to simulate intelligent behavior. Over the next four decades, despite many stumbling blocks, AI has grown from a dozen researchers, to thousands of engineers and specialists; and from programs capable of playing checkers, to systems designed to diagnose disease.

AI has always been on the pioneering end of computer science. Advanced-level computer languages, as well as computer interfaces and word-processors owe their existence to the research into artificial intelligence. The theory and insights brought about

by AI research will set the trend in the future of computing. The products available today are only bits and pieces of what are soon to follow, but they are a movement towards the future of artificial intelligence. The advancements in the quest for artificial intelligence have, and will continue to affect our jobs, our education, and our lives.

Evidence of Artificial Intelligence folklore can be traced back to ancient Egypt, but with the development of the electronic computer in 1941, the technology finally became available to create machine intelligence. The term artificial intelligence was first coined in 1956, at the Dartmouth conference, and since then Artificial Intelligence has expanded because of the theories and principles developed by its dedicated researchers. Through its short modern history, advancement in the fields of AI have been slower than first estimated, progress continues to be made. From its birth 4 decades ago, there have been a variety of AI programs, and they have impacted other technological advancements.

In 1941, an invention revolutionized every aspect of the storage and processing of information. That invention, developed in both the US and Germany was the electronic computer. The first computers required large, separate air-conditioned rooms, and were a programmers nightmare, involving the separate configuration of thousands of wires to even get a program running. The 1949 innovation, the stored program computer, made the job of entering a program easier, and advancements in computer theory lead to computer science, and eventually artificial intelligence. With the invention of an electronic means of processing data, came a medium that made AI possible.

Although the computer provided the technology necessary for AI, it was not until the early 1950's that the link between human intelligence and machines was really

observed. Norbert Wiener was one of the first Americans to make observations on the principle of feedback theory feedback theory. The most familiar example of feedback theory is the thermostat: It controls the temperature of an environment by gathering the actual temperature of the house, comparing it to the desired temperature, and responding by turning the heat up or down. What was so important about his research into feedback loops was that Wiener theorized that all intelligent behavior was the result of feedback mechanisms. Mechanisms that could possibly be simulated by machines. This discovery influenced much of early development of AI.

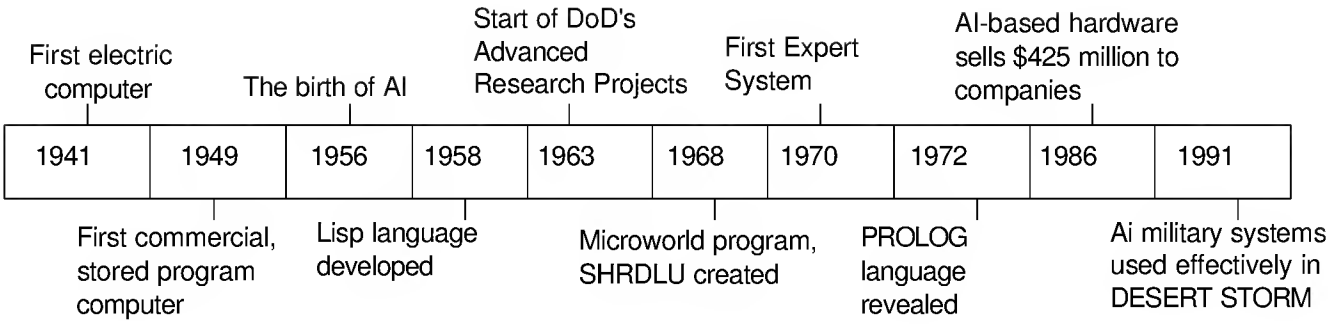
In late 1955, Newell and Simon developed *The Logic Theorist*, considered by many to be the first AI program. The program, representing each problem as a tree model, would attempt to solve it by selecting the branch that would most likely result in the correct conclusion. The impact that the logic theorist made on both the public and the field of AI has made it a crucial stepping stone in developing the AI field.

In 1956, John McCarthy regarded as the father of AI, organized a conference to draw the talent and expertise of others interested in machine intelligence for a month of brainstorming. He invited them to Vermont for "The Dartmouth summer research project on artificial intelligence." From that point on, because of McCarthy, the field would be known as “Artificial Intelligence”. Although not a huge success, the Dartmouth conference did bring together the founders of AI, and served to lay the groundwork for the future of AI research.

In the seven years after the conference, AI began to pick up momentum. Although the field was still undefined, ideas formed at the conference were re-examined. Centers for AI research began forming at Carnegie Mellon and MIT, and new challenges were

faced, such as creating systems that could efficiently solve problems by limiting the search, (Logic Theorist), and making systems that could learn by themselves. In 1957, the first version of a new program The General Problem Solver, GPS, was tested. The GPS was an extension of Wiener's feedback principle, and was capable of solving a greater extent of common sense problems. A couple of years after the GPS, IBM contracted a team to research artificial intelligence. In 1958, McCarthy announced his new development; the LISP language, which is still used today. LISP was soon adopted as the language of choice among most AI developers.

The military put AI based hardware to the test of war during Desert Storm. AI-based technologies were used in missile systems, heads-up displays, and other advancements. AI has also made the transition to the home. With the popularity of the AI computer growing, the interest of the public has also grown. Applications for the Apple Macintosh and IBM compatible computer, such as voice and character recognition have become available. Also AI technology has made steadying camcorders simple using "fuzzy logic". With a greater demand for AI-related technology, new advancements are becoming available. Inevitably Artificial Intelligence has, and will continue to affect our lives.



Timeline of major events of Artificial Intelligence, AI

4.2.3 Programming Languages

A vocabulary and set of grammatical rules for instructing a computer to perform specific tasks. The term programming language usually refers to high-level languages, such as BASIC, C, C++, COBOL, FORTRAN, Ada, and Pascal. Each language has a unique set of keywords (words that it understands) and a special syntax for organizing program instructions.

High-level programming languages, while simple compared to human languages, are more complex than the languages the computer actually understands, called machine languages. Each different type of CPU has its own unique machine language.
Lying between machine languages and high-level languages are languages called assembly languages. Assembly languages are similar to machine languages, but they are much easier to program in because they allow a programmer to substitute names for numbers.

Lying above high-level languages are languages called fourth-generation languages (usually abbreviated 4GL). 4GLs are far removed from machine languages and represent the class of computer languages closest to human languages.
Regardless of what language you use, you eventually need to convert your program into machine language so that the computer can understand it. There are two ways to do this:

- compile the program
- interpret the program

4.2.3.1 Assembler

This is programming language that is once removed from a computer's machine language. Machine languages consist entirely of numbers and are almost impossible for humans to read and write. Assembly languages have the same structure and set of commands as machine languages, but they enable a programmer to use names instead of numbers.

Each type of CPU has its own machine language and assembly language, so an assembly language program written for one type of CPU won't run on another. In the early days of programming, all programs were written in assembly language. Now, most programs are written in a high-level language such as FORTRAN or C. Programmers still use assembly language when speed is essential or when they need to perform an operation that isn't possible in a high-level language.

Assembler is used in many small robots where program and data memory are limited.

4.2.3.2 Basic

Basic is an acronym for Beginner's All-purpose Symbolic Instruction Code. Developed by John Kemeney and Thomas Kurtz in the mid 1960s at Dartmouth College, BASIC is one of the earliest and simplest high-level programming languages. During the 1970s, it was the principal programming language taught to students, and continues to be a popular choice among educators.

Despite its simplicity, BASIC is used for a wide variety of business applications. Microsoft's popular Visual Basic, for example, adds many object-oriented features to the standard BASIC.

Recently, many variations of BASIC have appeared as programming, or macro, languages within applications. For example, Microsoft Word and Excel both come with a version of BASIC with which users can write programs to customize and automate these application

Based on the BASIC language, Visual Basic was one of the first products to provide a graphical programming environment and a paint metaphor for developing user

interfaces. Instead of worrying about syntax details, the Visual Basic programmer can add a substantial amount of code simply by dragging and dropping controls, such as buttons and dialog boxes, and then defining their appearance and behavior.

Since its launch in 1990, the Visual Basic approach has become the norm for programming languages. Now there are visual environments for many programming languages, including C, C++, Pascal, and Java. Visual Basic is sometimes called a Rapid Application Development (RAD) system because it enables programmers to quickly build prototype applications.

A version of Basic, BASIC Stamp, is used to program the Boe-Bot used in this research project. The following is a sample screen for the Integrated Development Environment (IDE).

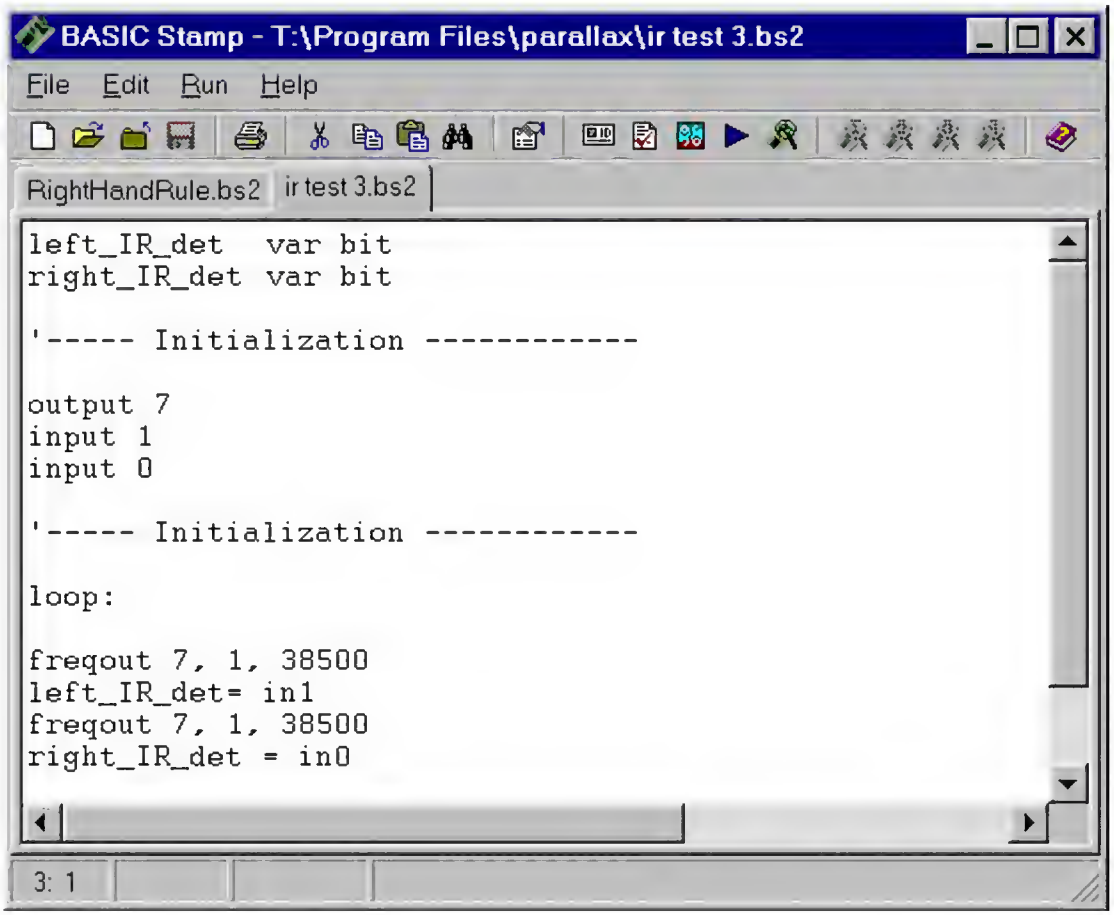


Figure 4. BASIC Stamp IDE Screen Shot

The BASIC Stamp programming language has limited variable and subroutine capabilities. It does have specialized functions and subroutines for use with the Basic Stamp II microprocessor.

4.2.3.3 C

C is a high-level programming language developed by Dennis Ritchie at Bell Labs in the mid 1970s. Although originally designed as a systems programming language, C has proved to be a powerful and flexible language that can be used for a variety of applications, from business programs to engineering. C is a particularly popular language for personal computer programmers because it is relatively small -- it requires less memory than other languages.

The first major program written in C was the UNIX operating system, and for many years C was considered to be inextricably linked with UNIX. Now, however, C is an important language independent of UNIX.

Although it is a high-level language, C is much closer to assembly language than are most other high-level languages. This closeness to the underlying machine language allows C programmers to write very efficient code. The low-level nature of C, however, can make the language difficult to use for some types of applications.

Most sophisticated commercial robots are programmed in C or C++.

4.2.3.4 C++

C++ is a high-level programming language developed by Bjarne Stroustrup at Bell Labs. C++ adds object-oriented features to its predecessor, C. C++ is one of the most popular programming language for graphical applications, such as those that run in Windows and Macintosh environments.

C++'s object oriented features make it easier to program robots that have many controls and components. Each control or component can be mirrored as a C++ object.

4.2.3.5 Java

A high-level programming language developed by Sun Microsystems, Java was originally called OAK, and was designed for handheld devices and set-top boxes. Oak was unsuccessful so in 1995 Sun changed the name to Java and modified the language to take advantage of the burgeoning World Wide Web.

Java is an object-oriented language similar to C++, but simplified to eliminate language features that cause common programming errors.

Java is a general purpose programming language with a number of features that make the language well suited for use on the World Wide Web. Small Java applications are called Java applets and can be downloaded from a Web server and run on your computer by a Java-compatible Web browser, such as Netscape Navigator or Microsoft Internet Explorer.

Java is being used in some commercial robots because of its portability and its ability to run applications that are downloaded to the robot.

4.2.3.6 Lisp

This is an acronym for ***list processor***, a high-level programming language especially popular for artificial intelligence applications. LISP was developed in the early 1960s by John McCarthy at MIT.

Lisp was used initially for robot programming because of its flexible list processing support and Lisp's dynamic programming support.

4.2.3.7 Prolog

This is a short for *Programming Logic,* Prolog is a high-level programming language based on formal logic. Unlike traditional programming languages that are based on performing sequences of commands, Prolog is based on defining and then solving logical formulas. Prolog is sometimes called a *declarative language* or a *rule-based language* because its programs consist of a list of facts and rules. Prolog is used widely for artificial intelligence applications, particularly expert systems.

Prolog was used for programming robots because of its backtracking capability commonly used in robotic-related algorithms.

Programming Table

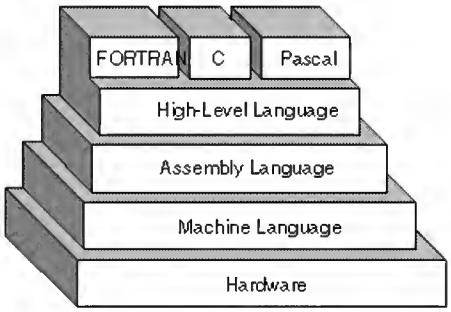


Figure 5. Programming Table

The figure shown above illustrates how the programming languages relate to the hardware. The hardware includes things like robot sensors, motors and other controls. Other high level languages like Java, BASIC, Lisp, Prolog and C++ would be positioned at the top of the hierarchy.

4.2.4 Robot Mechanics

Mechanics is the branch of physics concerning the motions of objects and their response to forces. Modern descriptions of such behavior begin with a careful definition of such quantities as displacement (distance moved), time, velocity, acceleration, mass, and force. Until about 400 years ago, however, motion was explained from a very different point of view. For example, following the ideas of the Greek philosopher and scientist Aristotle, scientists reasoned that a cannonball falls down because its natural position is in the earth. The sun, the moon, and the stars travel in circles around the earth because it is the nature of heavenly objects to travel in perfect circles.

Galileo showed that the speed of falling objects increases steadily during the time of their fall. This acceleration is the same for heavy objects as for light ones, provided air friction (air resistance) is discounted. The English mathematician and physicist Sir Isaac Newton improved this analysis by defining force and mass and relating these to acceleration.

4.2.4.1 Kinetics

Kinetics is the description of motion without regard to what causes the motion. Velocity (the time rate of change of position) is defined as the distance traveled divided by the time interval. Velocity may be measured in such units as kilometers per hour, miles per hour, or meters per second. Acceleration is defined as the time rate of change of velocity: the change of velocity divided by the time interval during the change. Acceleration may be measured in such units as meters per second per second or feet per second per second. Regarding the size or weight of the moving object, no mathematical

problems are presented if the object is very small compared with the distances involved. If the object is large, it contains one point, called the center of mass, the motion of which can be described as characteristic of the whole object. If the object is rotating, it is frequently convenient to describe its rotation about an axis that goes through the center of mass.

To fully describe the motion of an object such as a robot, the direction of the displacement must be given. Velocity, for example, has both magnitude (a scalar quantity measured, for example, in meters per second) and direction (measured, for example, in degrees of arc from a reference point). The magnitude of velocity is called speed.

Several special types of motion are easily described. First, velocity may be constant. In the simplest case, the velocity might be zero in which changes position would not change during the time interval. With constant velocity, the average velocity is equal to the velocity at any particular time. If time, t, is measured with a clock starting at $t = 0$, then the distance, d, traveled at constant velocity, v, is equal to the product of velocity and time.

$$d = vt$$

In the second special type of motion, acceleration is constant. Because the velocity is changing, instantaneous velocity, or the velocity at a given instant, must be defined. For constant acceleration, a, starting with zero velocity ($v = 0$) at $t = 0$, the instantaneous velocity at time, t, is

$$v = at$$

The distance traveled during this time is

$$d = \frac{1}{2} at^2$$

An important feature revealed in this equation is the dependence of distance on the square of the time. A heavy object falling freely (uninfluenced by air friction) near the surface of the earth undergoes constant acceleration. In this case the acceleration is 9.8 m/sec/sec (32 ft/sec/sec). At the end of the first second, a ball would have fallen 4.9 m (16 ft) and would have a speed of 9.8 m/sec (32 ft/sec). At the end of the second, the ball would have fallen 19.6 m (64 ft) and would have a speed of 19.6 m/sec (64 ft/sec).

Circular motion is another simple type of motion. If an object has constant speed but an acceleration that is always at right angles to its velocity, it will travel in a circle. The required acceleration is directed toward the center of the circle and is called centripetal acceleration. For an object traveling at speed, v, in a circle of radius, r, the centripetal acceleration is

Another simple type of motion that is frequently observed occurs when a ball is thrown at an angle into the air. Because of gravitation, the ball undergoes a constant downward acceleration that first slows its original upward speed and then increases its downward speed as it falls back to earth. Meanwhile the horizontal component of the original velocity remains constant (ignoring air resistance), making the ball travel at a constant speed in the horizontal direction until it hits the earth. The vertical and horizontal components of the motion are independent, and they can be analyzed separately. The resulting path of the ball is in the shape of a parabola.

4.2.4.2 Dynamics

To understand why and how objects accelerate, force and mass must be defined. At the intuitive level, a force is just a push or a pull. It can be measured in terms of either of two effects. A force can either distort something, such as a spring, or accelerate an object. The first effect can be used in the calibration of a spring scale, which can in turn be used to measure the amplitude of a force: the greater the force, F, the greater the stretch, x. For many springs, over a limited range, the stretch is proportional to the force

$$F = kx$$

where k is a constant that depends on the nature of the spring material and its dimensions.

4.2.4.3 Vectors

If an object is motionless, the net force on it must be zero. A book lying on a table is being pulled down by the earth's gravitational attraction and is being pushed up by the molecular repulsion of the tabletop. The net force is zero; the book is in equilibrium. When calculating the net force, it is necessary to add the forces as vectors.

4.2.4.4 Torque

For equilibrium, all the horizontal components of the force must cancel one another. This condition is necessary for equilibrium, but not sufficient. For example, if a person stands a book up on a table and pushes on the book equally hard with one hand in one direction and with the other hand in the other direction, the book will remain motionless if the person's hands are opposite each other. (The net result is that the book is being squeezed). If, however, one hand is near the top of the book and the other hand near the bottom, a torque is produced, and the book will fall on its side. For equilibrium

to exist it is also necessary that the sum of the torques about any axis be zero.

A torque is the product of a force and the perpendicular distance to a turning axis. When a force is applied to a heavy door to open it. If the force is exerted perpendicularly to the door and at the greatest distance from the hinges, a maximum torque is created. If the door were shoved with the same force at a point halfway between handle and hinge, the torque would be only half of its previous magnitude. If the force were applied parallel to the door (that is, edge on), the torque would be zero. For an object to be in equilibrium, the clockwise torques about any axis must be canceled by the counterclockwise torques about that axis. Therefore, one could prove that if the torques cancel for any particular axis, they cancel for all axes.

4.2.4.5 Newton's Three Laws of Motion

Newton's *first law of motion* states that if the vector sum of the forces acting on an object is zero, then the object will remain at rest or remain moving at constant velocity. If the force exerted on an object is zero, the object does not necessarily have zero velocity. Without any forces acting on it, including friction, an object in motion will continue to travel at constant velocity.

Newton's *second law of motion* relates net force and acceleration. A net force on an object will accelerate it—that is, change its velocity. The acceleration will be proportional to the magnitude of the force and in the same direction as the force. The proportionality constant is the mass, m, of the object.

$$F = ma$$

Acceleration, a, is measured in meters per second per second. Mass is measured in

kilograms; force, F, in Newton's. A Newton is defined as the force necessary to impart to a mass of 1 kg an acceleration of 1 m/sec/sec; this is equivalent to about 0.2248 lb.

A massive robot will require a greater force for a given acceleration than a small, light robot. What is remarkable is that mass, which is a measure of the inertia of an object (inertia is its reluctance to change velocity), is also a measure of the gravitational attraction that the object exerts on other objects. It is surprising and profound that the inertial property and the gravitational property are determined by the same thing. The implication of this phenomenon is that it is impossible to distinguish at a point whether the point is in a gravitational field or in an accelerated frame of reference. Einstein made this one of the cornerstones of his general theory of relativity, which is the currently accepted theory of gravitation.

Friction acts like a force applied in the direction opposite to an object's velocity. For dry sliding friction, where no lubrication is present, the friction force is almost independent of velocity. Also, the friction force does not depend on the apparent area of contact between an object and the surface upon which it slides. The actual contact area—that is, the area where the microscopic bumps on the object and sliding surface are actually touching each other—is relatively small. As the object moves across the sliding surface, the tiny bumps on the object and sliding surface collide, and force is required to move the bumps past each other. The actual contact area depends on the perpendicular force between the object and sliding surface. Frequently this force is just the weight of the sliding object. If the object is pushed at an angle to the horizontal, however, the downward vertical component of the force will, in effect, add to the weight of the object. The friction force is proportional to the total perpendicular force.

The left side of the equation is simply the net effective force. (Acceleration will be constant in the direction of the effective force). When an object moves through a liquid, however, the magnitude of the friction depends on the velocity. For most human-size objects moving in water or air (at subsonic speeds), the resulting friction is proportional to the square of the speed. Newton's second law then becomes

The proportionality constant, k, is characteristic of the two materials that are sliding past each other, and depends on the area of contact between the two surfaces and the degree of streamlining of the moving object.

Newton's *third law of motion* states that an object experiences a force because it is interacting with some other object. The force that object 1 exerts on object 2 must be of the same magnitude but in the opposite direction as the force that object 2 exerts on object 1. If, for example, a large adult gently shoves away a child on a skating rink, in addition to the force the adult imparts on the child, the child imparts an equal but oppositely directed force on the adult. Because the mass of the adult is larger, however, the acceleration of the adult will be smaller.

Newton's third law also requires the conservation of momentum, or the product of mass and velocity. For an isolated system, with no external forces acting on it, the momentum must remain constant. In the example of the adult and child on the skating rink, their initial velocities are zero, and thus the initial momentum of the system is zero. During the interaction, internal forces are at work between adult and child, but net external forces equal zero. Therefore, the momentum of the system must remain zero. After the adult pushes the child away, the product of the large mass and small velocity of the adult must equal the product of the small mass and large velocity of the child. The

momenta are equal in magnitude but opposite in direction, thus adding to zero.

Another conserved quantity of great importance is angular (rotational) momentum. The angular momentum of a rotating object depends on its speed of rotation, its mass, and the distance of the mass from the axis. When a skater standing on a friction-free point spins faster and faster, angular momentum is conserved despite the increasing speed. At the start of the spin, the skater's arms are outstretched. Part of the mass is therefore at a large radius. As the skater's arms are lowered, thus decreasing their distance from the axis of rotation, the rotational speed must increase in order to maintain constant angular momentum.

4.2.5 Robot Electronics

Electronics is the branch of physics that deals with the emission, behavior, and effects of electrons (as in electron tubes and transistors) and with electronic devices. Robot electronics are components used in robot sensors, central processing units and device controls.

The beginnings of electronics can be traced to various experiments with electricity. In the 1880s Thomas A. Edison and others observed the flow of current between elements in an evacuated glass tube. Edison, who was experimenting to improve his incandescent light bulb, added a second element near the filament in the tube. Under certain conditions, he noticed a bluish glow in the tube. This remained unexplained and unexploited for some years. In 1897 the British physicist J.J. Thomson determined that the Edison effect was due to the emission and passage of particles. Through experiment, he identified these particles as electrons. A short time later the English engineer John A. Fleming constructed the so-called thermionic valve (a two-electrode thermionic vacuum tube) by which an electrical current could be restricted to flow only in one direction. This rectifying process, also known as detection and demodulation, produced an output current that could be used to operate a telephone receiver or recording device. Fleming's valve was greatly improved in 1907 by the American engineer Lee De Forest, who added a

third element to the evacuated tube and produced the first triode, which could greatly amplify an electrical signal. From the triode the forms of the electron tube multiplied, giving rise to photomultiplier tubes, klystrons, magnetrons, and so forth.

In 1947 three scientists at Bell Laboratories--John Bardeen, William B. Shockley, and Walter H. Brattain--introduced the transistor, a simple device consisting of a small block of a semiconductor with three electrodes that could perform many of the functions of electron tubes. The invention of the transistor initiated a progressive miniaturization of electronic components that by the mid-1990s resulted in the manufacture of solid-state devices that contained more than 20,000,000 transistors on a single tiny silicon chip. Such high-density microcircuits, called microprocessors, have led to tremendous advances in computer technology and in computer-based automated systems (e.g., industrial robots and spacecraft-control systems). The availability of inexpensive microprocessors has also brought about the computerization of an enormous array of consumer products, ranging from self-focusing cameras and self-tuning televisions to programmable videocassette recorders and security systems. In addition, developments in optoelectronics (i.e., an approach that makes use of both optical and electronic phenomena) have yielded efficient photodetectors and solar cells, as well as light-emitting diodes, semiconductor lasers, and optical fibbers that are integral to many advanced communications systems.

4.2.5.1 Light Sensors

Light sensors can allow a robot behavior such as hiding in the dark. Light sensors can be purchased as photoresistors, or phototransistors. Photoresistors are similar to potentiometer except for a change in the resistance. The photoresistor is caused by change in light, but a turning of a knob causes a potentiometer. Phototransistors provide great sensitivity to light. A phototransistor is almost as easy to interface to a microscope as a photoresistor. Photodiodes possess great sensitivity, produce a linear signal over a very wide range of light levels, and respond rapidly to changes in illumination. This makes them useful in communication systems for detecting modulated light; the remote control receiver in almost every TV, stereo, and compact disc (CD) player on the market makes use of a photodiode. Photoresistors are usually made up of cadmium sulfide, or

CdS. Often photoresistors can be purchased at electronic stores. Other manufactures such as Texas Instruments, Hewlett-Packard and Motorola all have optoelectronic divisions that fabricate silicon photodiodes and phototransistors.

(a) Picture of a simple configuration using a phototransistor:

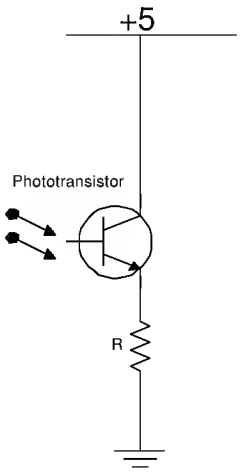


Figure 6. Phototransistor

4.2.5.2 Force Sensor

In mechanics, any action that tends to maintain or alter the position of a body or to distort it. The concept of force is commonly explained in terms of the three laws of motion set forth in his *Principia Mathematica* (1687). According to Newton's first principle, a body that is at rest or moving at a uniform rate in a straight line will remain in that state until some force is applied to it. The second law says that when an external force acts on a body, it produces an acceleration (change in velocity) of the body in the direction of the force. The magnitude of the acceleration is directly proportional to the magnitude of the external force and inversely proportional to the quantity of matter in the body. Newton's third law states that when one body exerts a force on another body, the second body exerts an equal force on the first body. This principle of action and reaction explains why a force tends to deform a body (*i.e.,* change its shape) whether or not it causes the body to move. The deformation of a body can usually be neglected when investigating its motion.

Because force has both magnitude and direction, it is a vector quantity and can be represented graphically as a directed line segment; that is, a line with a length equal to the magnitude of the force, to some scale, inclined at the proper angle, with an arrowhead at one end to indicate direction. The representation of forces by vectors implies that they are concentrated either at a single point or along a single line. This is, however, physically impossible. On a loaded component of a structure, for example, the applied force produces an internal force, or stress, that is distributed over the cross section of the component. The force of gravity is invariably distributed throughout the volume of a body. Nonetheless, when the equilibrium of a body is the primary consideration, it is generally valid as well as convenient to assume that the forces are concentrated at a single point. In the case of gravitational force, the total weight of a body may be assumed to be concentrated at its center of gravity Physicists use the newton, a unit of the International System (SI), for measuring force. A Newton is the force needed to accelerate a body weighing one kilogram by one meter per second per second. The formula $F = ma$ is employed to calculate the number of Newton's required to increase or decrease the velocity of a given body. In countries still using the English system of measurement, engineers commonly measure force in pounds. One pound of force imparts to a one-pound object an acceleration of 32.17 feet per second squared. Force sensors have proven the most reliable, exhibit the lowest noise, and produce the most easily interpreted signal of all sensors. Force sensors can be used to determine when the robot is in contact with another object and where that object is in relation to the robot. Such information allows the robot to maneuver away from collisions.

4.2.5.3 Sound Sensor

Sound sensors use sound waves to detect obstacles. Ultrasonic, high frequency sound waves, are normally used since they cannot be heard by humans. Sound waves are also used in water. SONAR uses this technique.

Ultrasonic range finders are used on robots and cameras.

4.2.5.4 Infrared Proximity Detectors

Near-infrared proximity detectors are often called IRs for short, but this term can be misleading. These detectors are insensitive to the long infrared wavelengths detected by pyroelectric sensors; rather, they are sensitive in the range just below the visible light, often around 880 nanometers (nm) wavelength. A near-infrared proximity sensor can be built from a Sharp detector and a near-infrared LED. The emitter is an LED made from gallium arsenide, which emits near- infrared energy at 880 nm. Both the emitters can be purchases from any semiconductor company that has optoelectronics divisions.

4.2.6 Examples of Robots

Robots come in a variety of forms and many different kinds of robots have been made in the past.

Photo 1. Urbie the Tactical Mobile Robot

The Defense Advanced Research Projects Association (DARPA) has enlisted JPL's Machine Vision Group in leading the design and implementation of its perception urban robot, a Tactical Mobile Robot. This urban robot (Urbie) is a joint effort of JPL, iRobot Corporation, the Robotics Institute of Carnegie Mellon University, and the University of Southern California Robotics Research Laboratory.

Urbie's initial purpose is mobile military reconnaissance in city terrain but many of its features will also make it useful to police, emergency, and rescue personnel. The robot is rugged and well-suited for hostile environments and its autonomy lends Urbie to many different applications. Such robots could investigate urban environments contaminated with radiation, biological warfare, or chemical spills. They could also be used for search and rescue in earthquake-struck buildings and other disaster zones.

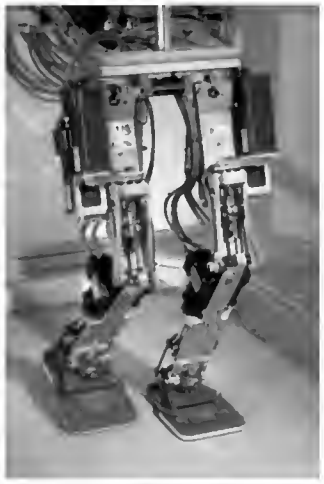


Photo 2. The BIP 2000 robot

The anthropomorphic approach to robots is attracting renewed interest for operation and service robotics because biped robots are able to move without particular adaptation to a human environment. The Bip 2000 robot, which will be presented at the Hanover world fair, June-October 2000, must be able to walk anthropomorphically in the absence of obstacles on a horizontal or slanted plane and to walk up and down stairs.

Bip 2000 constitutes a support for research, advanced experiments and demonstrations. Among the challenges taken up by the project are the automatic control of systems subjected to impacts and unilateral contact and integration techniques for the control system. A complete software environment for the specification, formal verification and automatic real-time programming of the control/command system was developed, using the ORCADD system whose robustness was greatly improved.

5. Testing

The object of these experiments is to get a self-contained, autonomous, mobile robot to navigate a maze from a starting point to an exit point. The robots were programmed to use basic maze navigation algorithms including Random Walk and Right Hand Rule. The Right Hand Rule is used in simple mazes, in which the robot follows the right side of the wall until it finds the finish. More advanced problem solving oriented navigation techniques like backtracking and machine learning will be used in the future.

The experiments used two very different robots: the WAO II and the Boe-Bot. Both robots were built from kits. The WAO II had a fixed design that limited its expandability. The Boe-Bot was designed with expandability in mind so multiple configurations were possible. Two configurations, Boe-Bot I and Boe-Bot II, were used in these experiments. The tests with the Boe-Bot II were not completed

The WAO II has two pressure sensors and a four-bit microprocessor that is limited in functionality and memory. It only supports about two dozen instructions. There are no variables that can be used to save information making conditional programming difficult.

On the other hand the, the Boe-Bot has a more powerful eight-bit microprocessor, the Stamp II. It uses a more powerful programming language, BASIC, and has inputs and outputs that can be connected to sensors that are mount on the robot. A patch board is included on the motherboard to make the addition of sensors, controls and support logic easier.

The Boe-Bot I is the Boe-Bot equipped with two infrared sensors. These sensors can detect obstructions in front of the Boe-Bot.

5.1 WAO II Robot

WOA II is an intelligent robot with a four-bit microcomputer. The robot is a hemisphere with a 26 punch-in keypad inserted on the back. WAO II can accomplish many things such as turn corners, use the Right Hand Rule, and do the random walk. Even though the WAO can do all those it can't do U-turns, and was not continuously checking for obstructions. This is all due to the lack of memory, which prevented a program to consist of more than 26 steps.

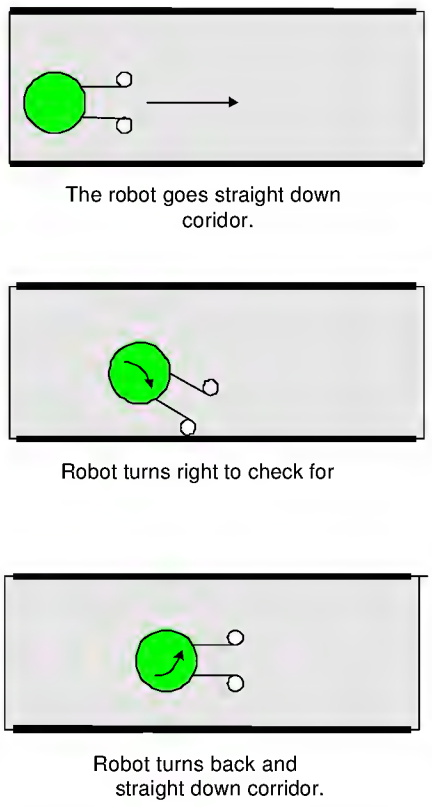


Figure 7. WAO II Navigating Corridor

5.1.1 Materials

Amount	Description
1	Robot body
2	Motors
2	Pressure sensors
3	AA Batteries
1	9-volt Battery
1	4-Bit Micro Computer with a 26 step memory
1	Programming keypad

Table 2. WAO II Materials List

5.1.2 Description of WAO II

Through 26 keys you can input the motion program. There are 8 movement command keys. Forward, back, stop, left turn, right turn, left pivot, right pivot, and buzzer. There are 4 program keys: if, "endif", for and next. "If" ~ "endif" controls the commands to the sensors. "For" and "next" controls the repeat conditions. The number 9 on the control pad is infinity.

There are three demonstration programs that are included with the kit.

1. Timer- delays the movement to start from 1- 9 minutes.
2. Dice- it will write one number from 1-6 randomly.
3. Roulette- it will randomly move rotating either left or right and will gradually slow down and stop.

Using a pen filler from a ballpoint pen and cutting it to size, WAO II is able to draw simple letters and designs.

Utilizing two sensors and wheels WAO II moves according to your program. Communication with a computer is also possible with the use of an optional interface card.

5.1.3 Test Procedures

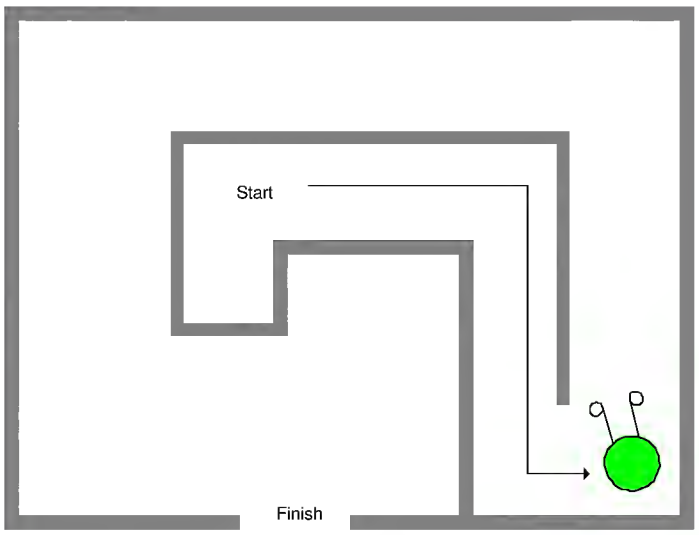


Figure 8. Simple Maze

A simple maze is a maze in which the robot can follow the right hand wall, also known as the Right Hand Rule, and find the finishing point.

The maze, in which the robot travels through, is a simple maze that is navigated using the Right Hand Rule. A simple maze compared to a complex maze is when there are a few right or left turns, and the robot can use the Right Hand Rule. On the other hand, a complex maze is one with dead ends, u- turns and many right or left turns. As you will see in the diagrams below the simple and complex mazes differ in a major variety of details.

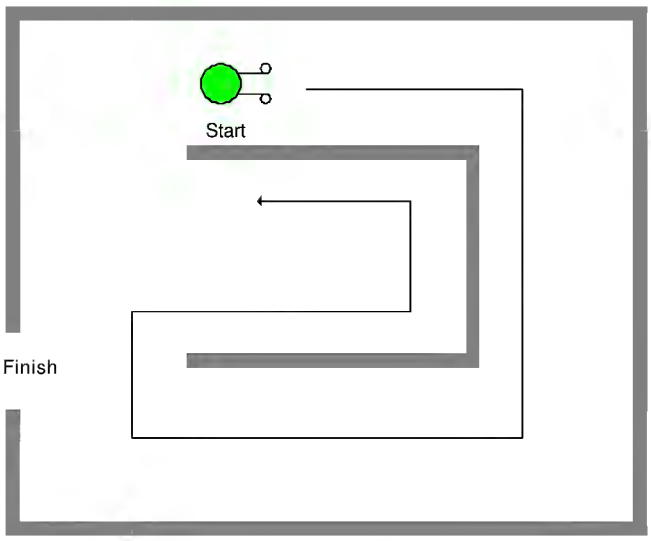


Figure 9. Complex Maze

A complex maze is a maze in which the robot can not get out of the maze using the Right Hand Rule. If the robot does follow the right wall it will just follow its own trail over and over again.

The following table lists the programs used with the WAO II.

Program	Description
1	Go Around A Rectangle
2	Right Hand Rule #1
3	Right Hand Rule #2
4	Maze Run #1
5	Maze Run #2

Table 3. WAO II Programs

5.1.4 Program One: Go around a rectangle

In the first program the robot does not use sensors but is told exactly when turning a corner. The robot goes twice around a rectangle a step at a time. If the robot did bump into the wall it would proceed with its instructions and run right into the wall.

```
For 2       ' repeat for two
 F9         ' forward for nine
 F9         ' forward for nine
 R3         ' right for three
 F3         ' forward for three
 R3         ' right for three
 F9         ' forward for nine
 R1         ' right for one
 F9         ' forward for nine
 R3         ' right for three
 F5         ' forward for five
 R3         ' right for three
 F1         ' forward for one
Next        ' next
```

Code Listing 1. WAO II Go Around A Rectangle

5.1.5 Program two: Right hand rule # 1

This program makes the robot goes around a square using sensors to detect when to turn a corner. The program does this by telling the robot to follow the right wall. This method is known as the Right Hand Rule. The robot can tell when to turn the corner because the right sensor is off.

```
For Infinity     ' repeat forever
   R1            ' right for one- check for wall
   S1            ' stop for one
   IF4           ' if right sensor is off then
                 '    turn the corner
      L1         ' left for one- straighten out
      F4         ' forward for four - move to
                 '    opening
      R1         ' right for one- turn corner
      For4       ' repeat four times
         If4     ' if right sensor is off then
                 '    continue turning right
            R1   ' right for one
         End If  ' end cycle
      Next       ' next
   End If        ' end cycle
   L1            ' left for one- line up with 'wall
   F1            ' forward for one- go forward
                 '    along wall
Next             ' next
```

Code Listing 2. WAO II Right Hand Rule #1

This first Right Hand Rule program only uses the right sensor. It ignores the left sensor. This can cause the robot to get stuck in a dead end or when a left turn is required. This program was tested with the robot going around the perimeter of a simple maze like a box.

5.1.6 Program Three: Right hand rule # 2

Program three adds five more stops. The stops gave the robot more accuracy because in program two the robot was not straight against the wall the whole time. When I added the stops in program three the robot stay on its course. Program three uses the right hand rule but if it gets stuck in a corner it would not be able to back its way out. This program uses the right sensor to detect its way around a maze.

```
For Infinity    ' repeat forever
   R1           ' right for one
   S1           ' stop for one
   If4/SR off   ' if right sensor is off turn
                '     - no more wall on right
      L1        ' left for one
      F3        ' forward for three
      S1        ' stop for one
      R1        ' right for one
      For4      ' repeat four times
         If 4   ' if right sensor is off then?
            S1  ' stop for one
            R1  ' right for one
         End If ' end of conditional turn
      Next      ' end of turn
   End If       ' end of right turn
   S1           ' stop for one
   L1           ' left for one
   S1           ' stop for one
   F1           ' forward for one
   S1           ' stop for one
Next            ' continue forever
```

Code Listing 3. WAO II Right hand rule # 2

5.1.7 Program four: Maze run # 1

In program four, unlike program three, the robot is able to get out of dead ends or corners. In this program the robot uses stops to keep itself on course. Program four uses sensors to find its way through a maze or around a square. Program four can get out of dead ends by backing up and using its sensors to find the right wall.

```
For infinity     ' repeat for infinity
   R1            ' turn right once to check wall
   S1            ' stop for one
   IF4           ' if right sensor is off then turn
                 'a corner
      L1         ' left for one- straighten out
      S1         ' stop for one
      F4         ' forward for four-move to
                 'opening
      S1         ' stop for one
      R1         ' right for one- begin to turn
                 '    corner
      S1         ' stop for one
      For4       ' repeat four times- turn corner
         IF4     ' if right sensor is off then
                 '    continue to turn corner
            R1   ' right for one
            S1   ' stop for one
         End If  ' end cycle
      Next       ' next
   End If        ' end cycle
   L1            'turn left for one- goto straight
   B1            ' go backwards for one- back up
                 'and turn left
   S1            ' stop for one
   TL3           ' turn left 90° for three
   S1            ' stop for one
   For2          ' repeat twice
    F1           ' forward for one- move along wall
    S1           ' stop for one
    L1           ' left for one
    S1           ' stop for one
   Next        ' next
 End If        ' end cycle
Next           ' next
```

Code Listing 4. WAO II Maze Run #1

This program checks the right sensors as a right turn is made to make sure that the robot does not continue to turn if the edge of a wall is found.

5.1.8 Program five: Maze run # 2

Program five adds four extra steps to program four. Both programs can back out of dead ends, use sensors, and use the right hand rule. The four steps added in program five gives it two extra stops, making the robot more straighter and makes the robot move forward two more spaces. Program five is the most complex program I have done with this experimental robot. The robot using this robot can use the right hand rule, sensors, stops, and backing out of dead ends.

```
For Infinity ' repeat forever
   R1          ' right for one - to check for wall
   S1          ' stop for one
   If4         ' turn corner if right sensor is off
      L1       ' left for one - straighten out
      S1       ' stop for one
      F4       ' forward for four - move to 'opening
      S1       ' stop for one
      R1       ' right for one-begin to turn 'corner
      S1       ' stop for one
      For4     ' repeat for 4 times-to turn corner
         If4   ' if right sensor is off then
               '   continue to turn corner
            R1 ' right for one
            S1 ' stop for one
         End If  ' end cycle
      Next     ' next
   End If      ' end cycle
   L1          ' left for one - goto straight
   S1          ' stop for one
   F1          ' forward for one- move along wall
   S1          ' stop for one
   If1         ' if right and left sensors are on
               '  check for dead end
      B1       ' backwards for one- backup and
               '  turn left
      S1       ' stop for one
      TL3      ' turn left 90° for three
      S1       ' stop for one
      For2     ' repeat for two
         F1    ' forward for one
         S1    ' stop for one
         L1    ' left for one
         S1    ' stop for one
      Next     ' next
   End If      ' end cycle
Next           ' next
```

Code Listing 5. WAO II Maze Run #2

This program uses an improved backup procedure when a dead end is detected. Smaller movements improve turning accuracy bit the code lacks sensor testing during the turn. The improvements come at the cost of slower movement.

PROGRAM	DESCRIPTION
Program one: Go around a rectangle	The robot goes around a rectangle twice using no sensors
Program two: Right Hand Rule #1	This program uses sensors to feel its way around the right hand side of the wall.
Program Three: Right Hand Rule # 2	This program uses sensors to feel its way around the right hand side of the wall plus adds more stops to keep it straight.
Program Four: Maze Run # 1	The robot is able to get out of dead ends by using its sensors and backing up. This program includes the right hand rule.
Program Five: Maze Run # 2	Program five can get out of dead ends, use sensors to follow the wall, turn corners and uses the right hand rule. It includes two more stops then program four.

5.1.9 Results

The WAO II was programmed using a number of maze programs plus a simple rectangle program. The latter showed that the robot had difficulty in turning accurately making repeatability difficult to attain. Small movements and turns were necessary to keep the accuracy at a reasonable level.

The average speed for forward movement in the maze programs was 1.8 in./sec. The pressure sensors used on the WAO II had a limited sensitivity of less than one inch. This meant the robot had to make small movements for the sensors to detect obstacles before the robot attempted to move the obstacle.

PROGRAM	**Results**
Program one: Go around a rectangle	The robot did not go around the rectangle by it self. But was programmed exactly how to get around the rectangle. The robot also needed help around corners.
Program two: Right Hand Rule #1	This program did not work sufficiently. It wandered away from wall constantly.
Program Three: Right Hand Rule # 2	Kept straight against wall. Turned corner but did not stay against wall but wandered away.
Program Four: Maze Run # 1	Kept going straight until it came to an end when it decided to do a u - turn instead of a 180 turn.
Program Five: Maze Run # 2	Kept going straight until dead end and successfully tuned around and went back to its starting place.

5.1.10 Analysis

WAO II is the control of my experiment since it is the first robot I tested my programs on. I made five different programs; one that is programmed to go around a square or rectangle, two programs which use the right hand rule and 2 that use sensors to detect the wall using the right hand rule and turn around at a dead end.

Program one was the simplest out of all five of the programs since the robot had all the directions computed into its memory. Program two through five all used sensors for the right hand rule and to turn corners. Except in program four and five the robot was programmed to use the right hand rule and to get out of a dead end.

5.1.11 Conclusion

The WAO II is a very restrictive robot, with only a 24 step memory there is no room for improvements. The robot can navigate through certain mazes, such as simple mazes. WAO II moves extremely slow through the maze and if you want to repeat the program over and over the robot must start in it original starting position to proceed through the maze. If the robot is not set in its original position the robot will perform differently and conclude with different conclusions. In conclusion, WAO II is accurate

Further testing could be done with this experiment by adding two extra sensors to the Boe-Bot robot. Further testing will also include to make more complicated programs for the robot to get through simple and complex mazes more quickly and efficiently.

5.2 Boe-Bot Robot With 2 Infrared Sensors

The Boe-Bot is based on a kit from Parallax Inc. The kit includes a chassis on which a Board of Education (BOE) motherboard is placed. The BOE contains a Basic Stamp II microprocessor and a plugboard that can be used for adding peripherals. This plugboard is used to add the two infrared sensors and transmitters.

5.2.1 Material

The following materials were used to build and program the Boe-Bot.

Number of parts	Parts
1	Boe-Bot chassis
1	Board of Education (BOE)
1	Battery pack
2	Servos
2	Plastic wheels
1	Polyethylene ball
2	9/32" Rubber Grommet
1	13/32" Rubber Grommet
2	O-ring tires
1	Cotter pin
10	4-40 locknuts
2	4-40 flathead screws
8	3/8" 4-40 screws
8	1/h" 4-40 flathead screws
4	1/2" Standoffs
1	Serial Cable
4	AA alkaline batteries
1	Parallax Cd

Table 4. Boe-Bot Table of Materials

The BOE contains a Basic Stamp II microprocessor. The microprocessor contains 2048 bytes of nonvolatile, erasable program memory and 256 bits of volatile data memory. The microprocessor is programmed using the Basic Stamp IDE found on the Parallax CD. The IDE runs on a Windows 98 PC.

5.2.2 Description

Boe-Bot runs on two servos, which turn the wheels of the robot. When the right servo runs forward and the left servo runs backwards the robot turns left, if the right servo rotates backwards and the left forwards then the robot will turn left. An example of a vehicle that moves like this is an army tank.

By shining infrared into the Boe-Bot's path and looking for its reflection, object, such as walls, can be detected. Infrared LED, Light Emitting Diodes, circuits are used to send a 38.5 kHz, kilohertz, signal by using a unique property of freqout. This allows you to control a harmonic of the frequots PWM wave shaping signal via IR LED circuits.

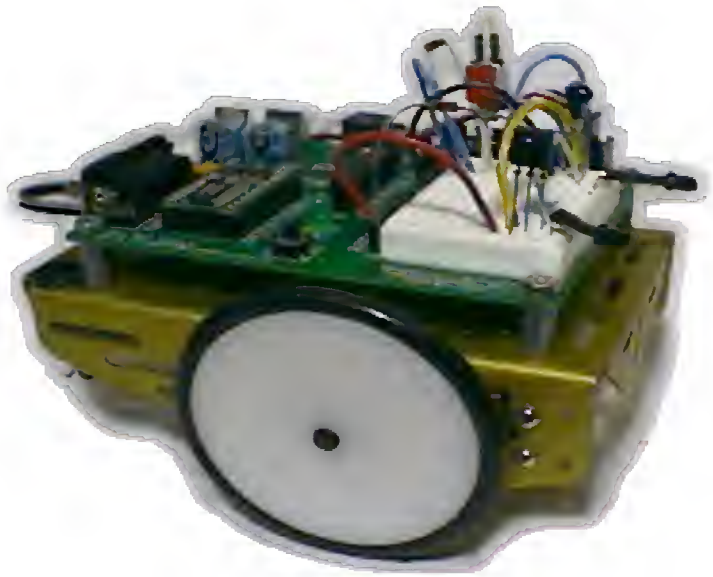


Photo 3. Boe-Bot with 2 infrared sensors

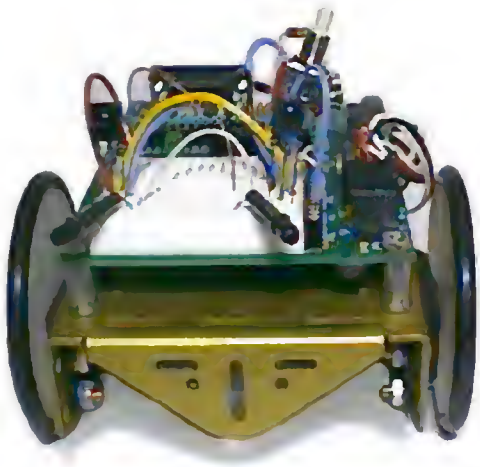


Photo 4. Boe-Bot front view

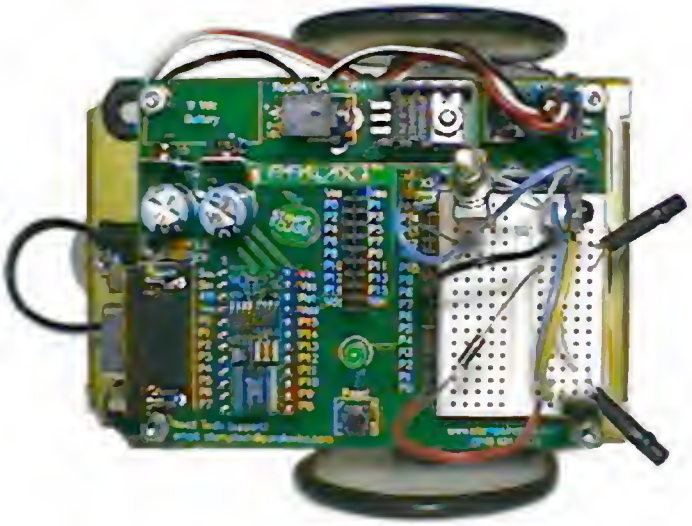


Photo 5. Top view of Boe-Bot

5.2.3 Construction

Mounting the Topside Hardware

Figure 1.22 shows the Boe-Bot chassis, topside hardware and mounting screws.

Parts List:

(1) Boe-Bot Chassis
(4) Standoffs
(4) 1/4" 4-40 screws
(2) 9/32" Rubber grommets
(1) 13/32" Rubber grommet

Photo 6. Chassis and topside hardware

Photo 7. Topside hardware assembled

Photo 8. Battery pack and mounting hardware

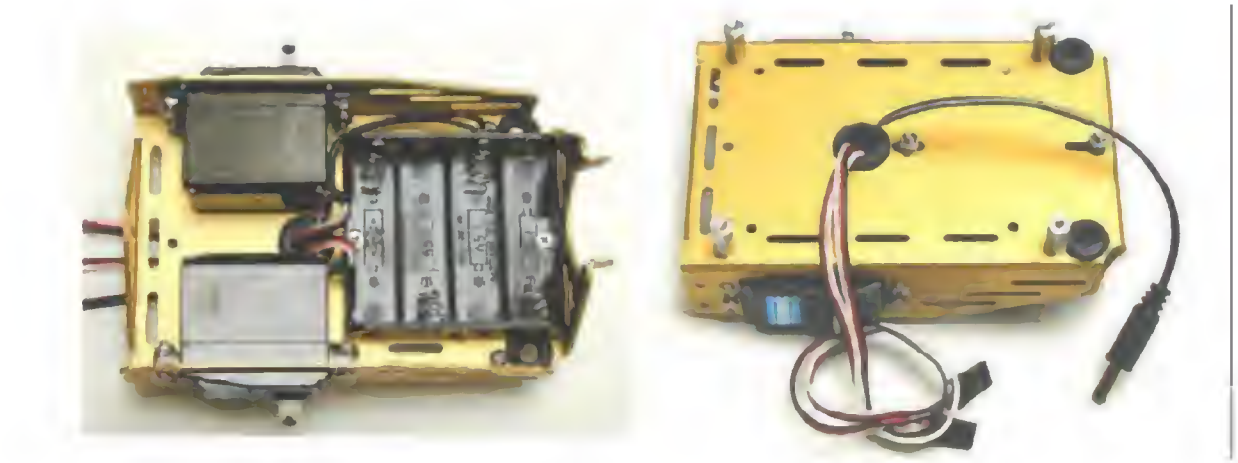


Photo 9. Battery pack are installed and wires are pulled through bottom of chassis

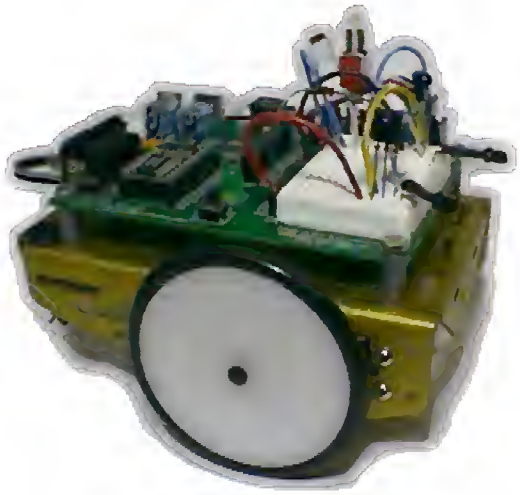


Photo 10. Board of Education and wheels are attached to the chassis

The Board of Education (BOE) is the circuit board on top of the chassis. This is the final construction step. The Boe-Bot is ready to be programmed.

5.2.4 Test Procedures

The Boe-Bot has a program downloaded from the Basic Stamp IDE through a serial port on a Windows 98 PC to the BOE on the Boe-Bot. The program is stored in the

non-volatile program memory. On-board battery power or an external power supply can be used to power the Boe-Bot during the download.

The Boe-Bot can then be disconnected from the PC. Battery power is then used to run a program when the Boe-Bot runs through a maze.

Prior to running the maze programs, the Boe-Bot must be calibrated. The infrared sensors and the servo motors must be calibrated. The following programs were used to perform these tasks.

```
left_IR_det  var bit
right_IR_det var bit

'----- Initialization ------------

output 2
output 7
output 1

'----- Initialization ------------

loop:

freqout 7, 1, 38500
left_IR_det= in8

freqout 1, 1, 38500
right_IR_det = in0

debug home, "Left=" , bin1 left_IR_det, " Right=
",bin1 right_IR_det, "   "


goto loop
```

Code Listing 6. Boe-Bot IR Test Program

In this program, the infrared sensors were tested to see how close the robot can come to an object, such as a wall, before the IR sensor detects the reflected light from the object. The IR transmitter sends out an expanding cone of light (see figure below). The two cones from the transmitters overlap in front of the center of the robot. If an obstruction is not directly in front then only one of the detectors will recognize the object.

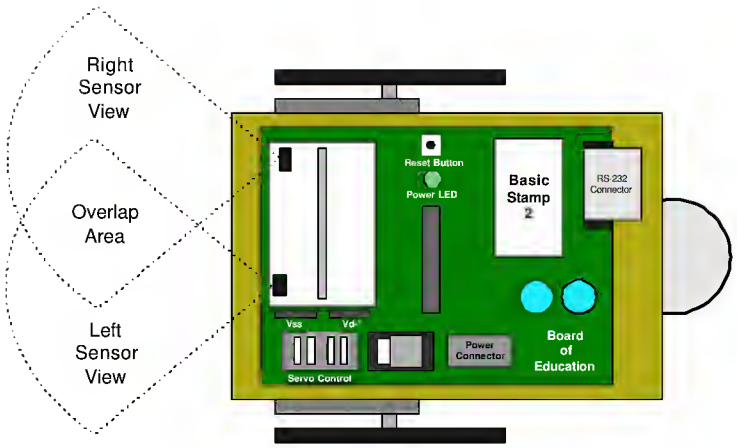


Figure 10. Boe-Bot IR Transmitter Light Cones

This program tests the IR transmitter and receivers. A blank white note card is held in front of each sensor and slowly moved away until the sensor no longer detects the reflected IR light. The note card is then brought closer to the sensor to verify that the IR detector can still detect the light when the card is close enough. This test indicates how close the robot will come to an obstruction assuming it can react as soon as an object is detected.

```
'Speed test program


speed      var word
puls12     var word
puls13     var word
direction    var nib
base       con 800
ctr      var word

low 12
low 13

speed= 30

direction= 1
gosub sendpulse
direction= 2
gosub sendpulse

idleit:
  direction = 0
  gosub sendpulse
  goto idleit

' Send pulses for one second

sendpulse:

for ctr = 1 to 50
  gosub getpuls
  pulsout 12, puls12
  pulsout 13, puls13
  pause 20
next

return


getpuls:
  branch direction,
[stopit,forward,back,left,right]

left:
  puls12= base + speed
  puls13= base + speed
  return

right:
  puls12= base - speed
  puls13= base - speed
  return

stopit:
```

```
  pulse12= base
  pulse13= base
  return

forward:
  pulse12 = base + speed
  pulse13 = base - speed
  return

back:
  pulse12= base - speed
  pulse13= base + speed
  return
```

Code Listing 7. Boe-Bot Servo Motor Calibration

This program, the left and right servos motors are calibrated. When the base is set at 800 (as seen below) both servos stay still and the robot does not move. When the speed has been adjusted the servo motor is controlled to move backwards or forward. This program is one of the few that do not use the IR sensors.

```
' Walk a square

speed      var word
puls12     var word
puls13     var word
direction    var nib
base       con 800
ctr      var word
sqrctr     var word
sqrdir     var word
sqrturn var word
sqrside var word
sidectr var word

low 12
low 13

speed= 30

' 0 = stop, 1 = forward, 2 = back, 3 = left, 4 =
right
sqrdir = 1
sqrturn = 3
sqrside= 2

gosub square
sqrdir= 2
sqrturn= 4
gosub square



idleit:
  direction = 0
  gosub sendpulse
  goto idleit

' Send pulses for one second

sendpulse:

for ctr = 1 to 50
  gosub getpuls
  pulsout 12, puls12
  pulsout 13, puls13
  pause 20
next

return


getpuls:
  branch direction,
[stopit,forward,back,left,right]
```

```
left:
  puls12= base + speed
  puls13= base + speed
  return

right:
  puls12= base - speed
  puls13= base - speed
  return

stopit:
  puls12= base
  puls13= base
  return

forward:
  puls12 = base + speed
  puls13 = base - speed
  return

back:
  puls12= base - speed
  puls13= base + speed
  return

square:

for sqrctr = 1 to 4
  for sidectr = 1 to sqrside
    direction= sqrdir
    gosub sendpulse
  next
  direction= sqrturn
  gosub sendpulse
next
return
```

Code Listing 8. Boe-Bot Square

The square program repeatedly circumscribes a square. If the robot is calibrated accurately then it should be able to cover the same ground each time it repeats the process.

```
'Random walk with IR dectors

'----- Declarations --------------

maxpulses    con 50
base       con 800
speed      var word
forwardspeed var word
direction var nib

pulse_count var byte           ' For...next loop
counter.
left_IR_det var bit            ' Two bit variables
for saving IR
right_IR_det var bit           ' detector output
values.

btnWk      var  byte

'----- Initialization ------------

btnWk = 0

output 2           ' Set all I/O lines sending
freqout
output 7           ' signals to function as outputs
output 1
freqout 2, 2000, 3000          ' Declare a
variable for counting.
low 12           ' Set P12 and 13 to output-low.
low 13


'----- Main Routine --------------
speed=30
forwardspeed= 50

waitloop:          ' Wait until button pressed to
start roaming
   BUTTON 15,0,255,255,btnWk,0,waitloop
main:
   BUTTON 15,0,255,255,btnWk,1,waitloop


             ' Detect object on the left.
freqout 7, 1, 38500            ' Send freqout
signal - left IRLED.
left_IR_det = in8              ' Store IR detector
output in RAM.
             ' Detect object on the right.
freqout 1, 1, 38500            ' Repeat for the
right IR pair.
right_IR_det = in0
```

```
              ' With the exception that values
stored in RAM are used instead of
              ' input register values, the decision
making process is the same as
              ' the one used in Program Listing
3.2.

if left_IR_det = 0 and right_IR_det = 0 then
u_turn
if left_IR_det = 0 then right_turn
if right_IR_det = 0 then left_turn
              ' The commands from this point onward
are identical to
              ' Program Listing 3.2: Roaming with
Whiskers.
forward:          ' If no detect, one forward
pulse.
debug "Forward", cr
pulsout 12, base - forwardspeed
pulsout 13, base + forwardspeed
pause 20
goto main             ' Check again.


'----- Navigation Routines -------

left_turn:            ' Left turn routine.
debug "Turn left", cr
gosub backward            ' Call Backward: before
turning.
for pulse_count = 0 to maxpulses
pulsout 12, base + speed
pulsout 13, base + speed
pause 20
next
goto main

right_turn:           ' Right turn routine.
debug "Turn right", cr
gosub backward            ' Call Backward: before
turning.
for pulse_count = 0 to 30
pulsout 12, base - speed
pulsout 13, base - speed
pause 20
next
goto main

u_turn:             ' U-turn routine.
debug "U-Turn", cr
gosub backward            ' Call Backward: before
turning.
gosub left_turn
gosub left_turn
```

```
goto main

'----- Navigation Subroutine -----

backward:              ' Used by each navigation
routine.
debug "Backward", cr
for pulse_count = 0 to 60
pulsout 12, base + speed
pulsout 13, base - speed
pause 20
next
return
```

Code Listing 9. Boe-Bot Random Walk

The random walk program uses the infrared detectors to locate obstacles. It backs away from an obstacle and heads off in a different direction.

5.2.4.1 Right Hand Rule

This section presents the Right Hand Rule program starting with a state diagram.

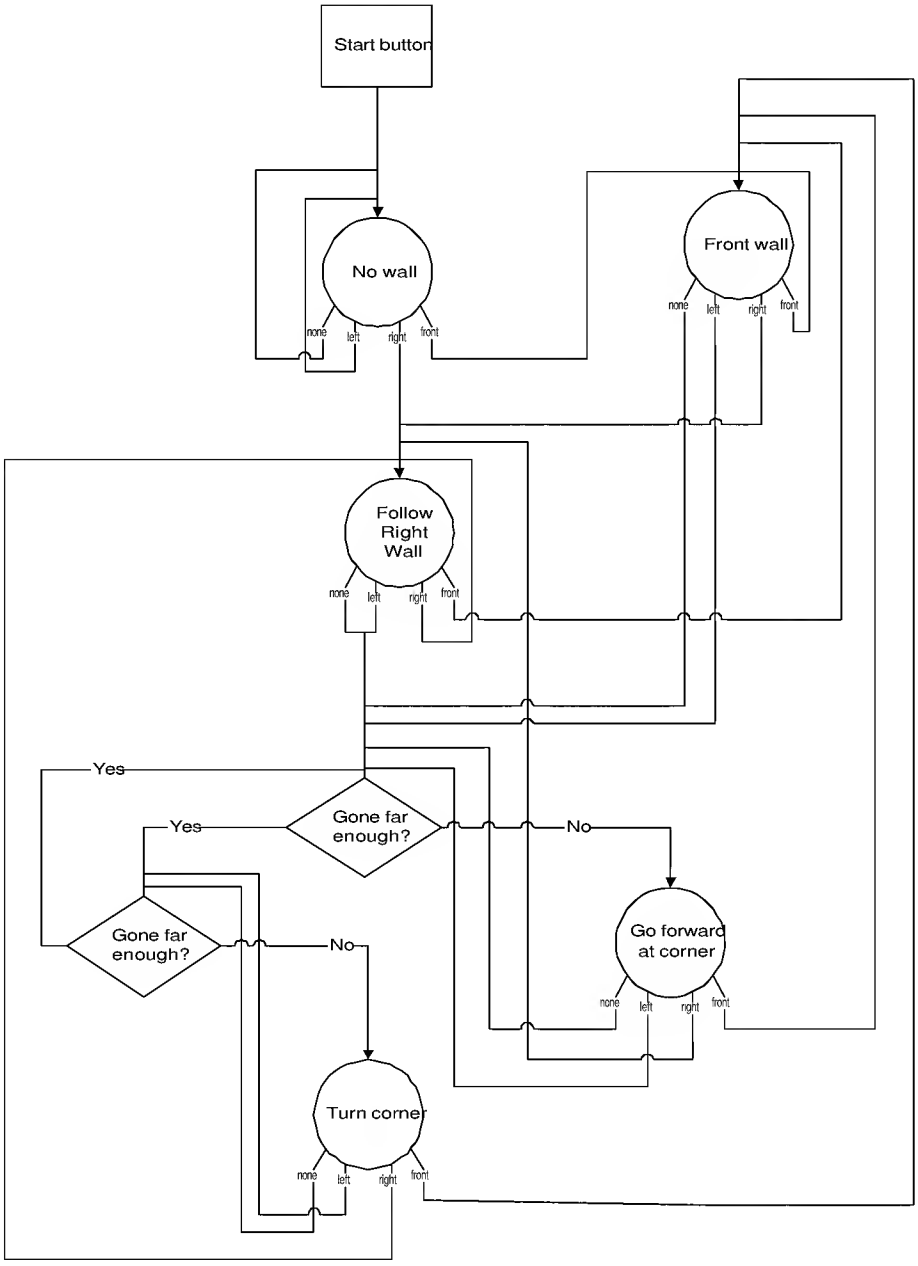


Figure 11. Right Hand Rule State Diagram

The box, Start button, is the action of the start button on the robot being pressed.

The initial state assumes the robot does not detect a wall. If the robot detects a left wall or

none at all then the robot changes state and continues forward while it detects a right wall. This is where the robot starts using the Right Hand Rule.

There is a state that handles the condition where the robot sees a wall in front of it. This condition is checked in all other states. The robot will pivot left until another condition is met. This may mean a 90° turn or even a 180° turn. The result should leave the robot with a wall to its right. The program then changes state depending upon the condition that ends this state.

The robot continues to follow the Right Hand Rule with the wall detected to the right. It detects a corner when it no longer detects a wall on the right. This consists of two possible conditions: no wall in front or a wall only on the left. The robot changes to the state that continues forward so the body of the robot will clear the corner.

The robot then changes state to turn right. The right turn continues until the robot turns the corner or detects a wall on the right or in front thereby continuing the use of the Right Hand Rule.

Using the Right Hand Rule states over and over allow the robot to successfully navigated a simple maze. The right turn and front wall support addresses the robot drifting left or right away from or towards the right wall it is following. The right turn state has a maximum turning radius but not a minimum. The robot later is controlled by the sensors. The front wall support would be used if the robot turns towards the wall and both sensors detect the wall.

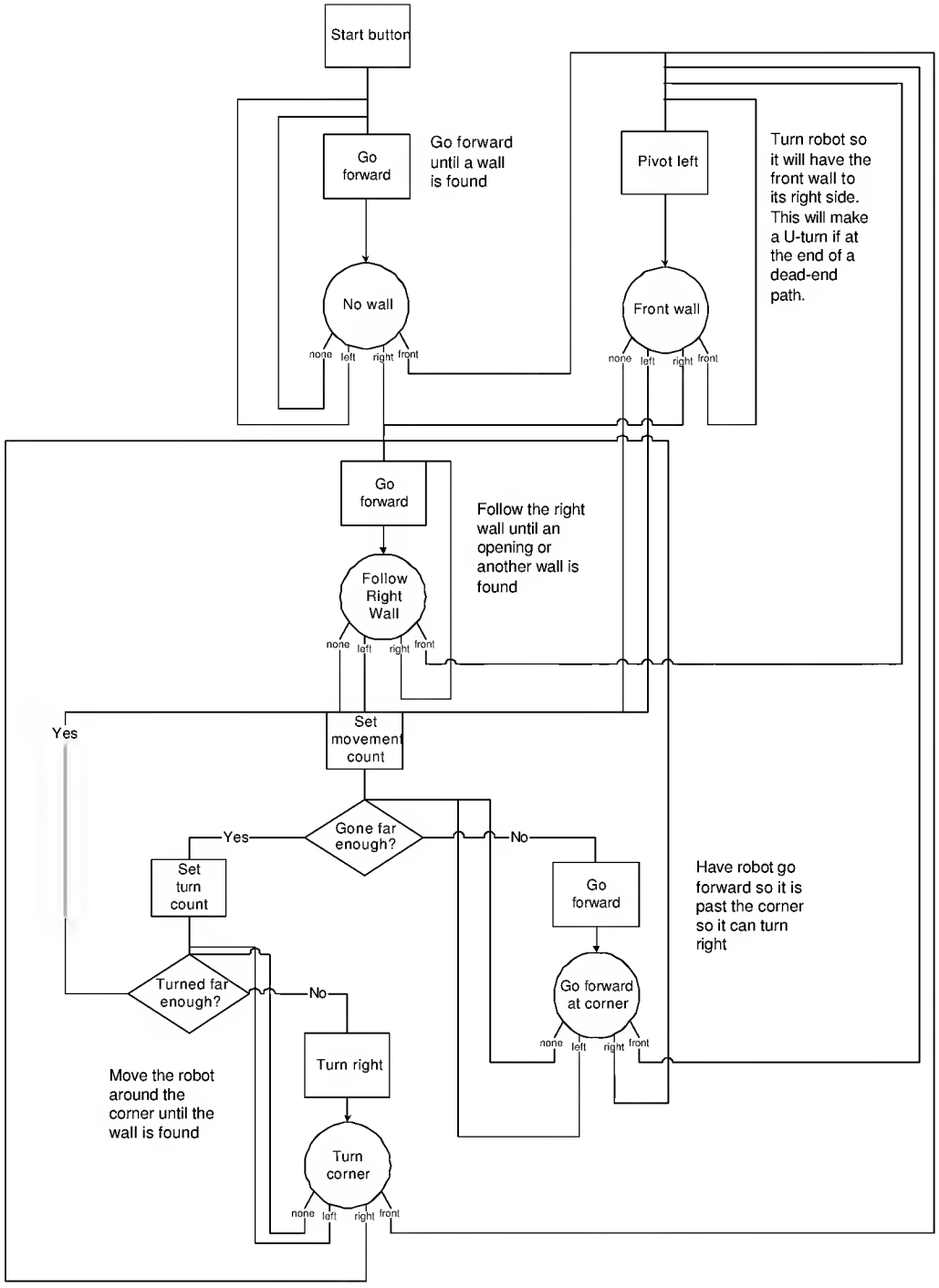


Figure 12. Right Hand Rule Flowchart

The flowchart is similar to the state chart, but the flowchart adds specific actions such as move forward, pivot left and turn right. The setting of the movement counts are included in the flow chart making the flowchart match the program.

```
'Righthand rule

'----- Declarations --------------

maxpulses    con 50             ' servo motor
control constants and variables
base       con 800
normalspeed  con 30
forwardLeft  con 60             ' may be different
from forwardRight due to motor adjustment
forwardRight con 60

speedLeft var word
speedRight   var word

pulse_count  var byte           ' For...next loop
counter.
pulse_limit  var byte

sensors var nib   ' sensor data storage

IR_right  var sensors.bit0
IR_left var sensors.bit1

btnWk      var  byte       ' button working
variable


' ----   wallState  -----

movePastCornerCount       con  30
turnRightCornerCount      con  60
maxTurn      con  4


moveCount var byte
rotateCount  var nib


'----- Initialization ------------

btnWk   = 0         ' initialize button

sensors = 0         ' initialize sensor status

output 7            ' setup IR LED output
output 6            ' setup status LED output
low 6               ' turn off LED

input 1             ' IR sensors
input 0

low 12              ' Set P12 and 13 to output-low.
low 13              '    for motor control
```

```
'----- Main Routine --------------

waitloop:           ' Wait until button pressed to
                    ' start roaming
   BUTTON 15,0,255,255,btnWk,0,waitloop
   goto enterMainLoop


'---- Main loop, no wall found yet -----

mainWallOnLeft:
mainNoWall:
   'debug home, "No Wall In Front   ", cr
   gosub moveForward       ' move forward

enterMainLoop:
   gosub checkSensors      ' get new speed
depending
                           ' upon sensors
   branch sensors, [mainWallInFront,
mainWallOnLeft, mainWallOnRight, mainNoWall]


'-- Wall in front, turn left until wall on right

mainWallInFront:
   'debug home, "Wall In Front      ", cr
   gosub pivotLeft              ' turn left in
place
   gosub checkSensors      ' get new speed
depending
                           ' upon sensors

   branch sensors, [mainWallInFront,
movePastCorner, mainWallOnRight, movePastCorner]


'---- Wall on right ----
'
' Look for opening on right
' Look for wall in front

mainWallOnRight:
   'debug home, "Wall On Right     ", cr
   gosub moveForward       ' move forward
   gosub checkSensors      ' get new speed
depending
                           ' upon sensors

   branch sensors, [mainWallInFront,
movePastCorner, mainWallOnRight, movePastCorner]
```

```
'---- Wall was on right, moving foward to clear
'----  corner if no wall seen

movePastCorner:
   moveCount = movePastCornerCount

movePastCornerLoop:
   if moveCount = 0 then turnRightCorner

   high 6          ' turn on status LED
   moveCount = moveCount - 1

   'debug home, "Moving past corner=", dec
   '  moveCount, "       ", cr
   gosub moveForward       ' move forward
   gosub checkSensors      ' get new speed
depending
                           ' upon sensors
   low 6                   ' turn off LED
   branch sensors, [mainWallInFront,
movePastCornerLoop, mainWallOnRight,
movePastCornerLoop]


'---- Wall was on right, moved foward, turn right
'---- until wall seen or turn count exhausted

turnRightCorner:
   moveCount = turnRightCornerCount
turnRightCornerLoop:
   if moveCount = 0 then mainNoWall


   moveCount = moveCount - 1

   high 6          ' turn on status LED
   'debug home, "Turning corner=", dec moveCount,
   '  "       ", cr
   gosub turnRight              ' turn right
   gosub checkSensors      ' get new speed
depending
                           ' upon sensors
   low 6           ' turn off LED
   branch sensors, [mainWallInFront,
turnRightCornerLoop, mainWallOnRight,
turnRightCornerLoop]


'===== Movement subroutines ====
'
'----- Move robot -----

moveForward:
   'debug "go forward ", cr
```

```
   speedLeft    = base - forwardLeft
   speedRight   = base + forwardRight
   goto moveRobot

turnRight:
   'debug " turn right", cr
   speedLeft = base - (normalspeed/2)
   speedRight   = base + (normalspeed*2)
   goto moveRobot

turnLeft:
   'debug " turn left", cr
   speedLeft = base + (normalspeed/2)
   speedRight   = base - (normalspeed*2)
   goto moveRobot

pivotRight:
   'debug " pivot right", cr
   speedLeft = base + normalspeed
   speedRight   = base + normalspeed
   goto moveRobot

pivotLeft:
   'debug " pivot left ", cr
   speedLeft = base - normalspeed
   speedRight   = base - normalspeed

moveRobot:
   pulsout 12, speedLeft  ' move robot
   pulsout 13, speedRight
   pause 20
   return


'----- Check IR sensors  ----
'  00 - left and right
'  01 - left only
'  10 - right only
'  11 - neither

checkSensors:

   freqout 7, 1, 38500
   IR_left= in1
   freqout 7, 1, 38500
   IR_right = in0

   'debug "Sensors=", bin2 sensors, cr
   'return
```

Code Listing 10. Boe-Bot Right Hand Rule

Right Hand Rule program was actually easier to write than the one used with the WAO II because the sensors are better and variables can be used to check both sensors at the same time. Variables can also be used to keep track of actions that are being performed such as a turn. Subroutines also eliminate duplicating operations in the program. The branch statement allows straightforward translation from the state diagram and flowchart to the program.

The program starts by waiting for a button press making it easier to position the robot at a starting point. It then follows the same logic as the flowchart by going straight ahead until a wall is found. Each of the branch statements matches the state change condition in the flowchart.

Debug statements are commented out to improve speed. They are only useful when the robot is tethered to the PC. The LED attached to pin 6 is used to keep track of when the right turn states are being executed.

Each movement action like the moveForward subroutine only moves the robot a small distance. Movement over a greater distance is done through repeated calls to the subroutines. This combined with repeated checking of the sensors allows the robot to make adjustments very quickly.

The same states used in the flowchart and state diagrams are used in the program. The robot turns left whenever an obstacle is found in front so the robot can follow along the right side of the wall. A right turn is made when the wall ends.

The techniques used with the Boe-Bot could not be used with the WAO II due to insufficient programmable memory.

5.2.5 Results

Five test programs were used with the Boe-Bot. Two were used for calibration. One, the Square program, was used to test the accuracy of the servo motors. The random walk and right hand rule programs were run repeatedly.

The Boe-Bot was able to sustain 4 in./sec. while detecting a wall. This was not the robot's fastest speed but it was not tested at a faster rate. The infrared sensor range was 6 to 8 inches depending upon the angle.

PROGRAM	**Results**
IR Testing	This test indicated how close the robot came to an obstruction assuming it can react as soon as an object is detected.
Sensor Calibration	In this test, both the left and right servo's motors were calibrated. When the base was set at 800, both servos stood still and the robot did not move. When the speed was adjusted, the servo motor is controlled to move backwards or forward
Square	The robot went around the square box successfully using the infrared sensors to detect the walls of the box.
Random Walk	The random walk was fast and easy but a default in the program was that the robot sometime did not get out of the maze successfully.
Right hand Rule	Regularly navigated simple mazes. Repeatability is good even when the robot was not started at exactly the same position or orientation. It follows the wall fairly well but it can get lost on occasion. Further refinements are necessary.

Table 5. Boe-Bot Programs

5.2.6 Analysis

Each of the programs worked they way they were programmed. The IR program was used for testing how close the robot can come to the wall before detecting it. In the second program the sensors were calibrated to travel forwards, backwards, left and right. The third and final program is the most important. This program lets the robot navigate through a simple maze using the two IR sensors. This program also allows the robot to do U-turns, turn corners, and use the Right Hand Rule.

5.2.7 Conclusion

The Boe-Bot's speed and sensor accuracy was superior to the WAO II. The infrared sensors had a longer range and did not interfere with the movement of the Boe-Bot. The Boe-Bot was not run at its top speed.

The Boe-Bot's random walk was more effective than the WAO II because the Boe-Bot's did not come in contact with an obstacle but it failed to consistently navigate a maze. The Boe-Bot detected walls well but the Right Hand Rule program had difficulty in moving parallel to a wall.

Future research using a backtracking algorithm should improve performance.

6. Future Work

The WAO II will be abandoned because of its limited capacity and performance. The Boe-Bot has plenty of room for growth and is suitable for adding more sophisticated programs. Adding more infrared detectors will also be done to allow the Boe-Bot to track the edge of a wall more accurately.

7. Bibliography

http://www.askjeeves.com/main/metaanswer.asp?metaEngine=directhit&origin=0&Meta URL.html.

Force, http://www.britannica.com, Britannica.com Inc., 2000.

Robot Mechanics, http://www.britannica.com, Britannica.com Inc., 2000.

Algorithm, http://webopedia.internet.com/Programming/algorithm.html .

Artificial Intelligence, http://webopedia.internet.com

AI, http://library.thinkquest.org/2705/history.html.

Bip, http://www.inria.fr/Equipes/BIP-eng.html.

Sincell, Marc, Science Magazine News, *Robotics: Rescue Droids Stumble In an Urban Jungle*, Aug., Vol. 289, Austin, Texas, 2000.

Hara, Yoshiko, EE Times, *Robot-San Steps Up,* Jan., Vol. 1149, 2001.

Druin, Allison and Hendler, James, Robots for kids *Exploring New Technologies for Learning,* Morgan Kaufmann, San Francisco, CA, 2000.

Jones L. Joseph, Seiger A. Bruce, Flynn M. Anita, Mobile Robots *Inspiration to Implementation*, Second Edition, Natick, Ma, 1999.

Parallax Inc., *Basic Stamp Manual*, Version 1.9, PIC, 1998.

Williams, Al, *Microcontroller Project with Basic Stamps*, R&D Books, Lawrence, Kansas, 2000.

Wang Wallace, *Visual Basic 6 for Dummies*, Transworld, Foster City, CA, 1998.

Grolier Incorporated, Danbury, Connecticut, *The New Book Of Knowledge*, Volume 3, C, Grolier Incorporated, 1988.